Análisis de Experimentos de Comunicaciones de Tiempo Real en el Bus CAN

Fernando G. Tinetti[1], Fernando L. Romero, Alejandro D. Pérez

Instituto de Investigación en Informática LIDI (III-LIDI)
Facultad de Informática, UNLP, 50 y 120, La Plata, Argentina
{fernando, fromero}@lidi.info.unlp.edu.ar, perez.alejandrodaniel@gmail.com

Resumen. Este trabajo presenta el desarrollo de un análisis de tiempos de comunicaciones en CAN, inicialmente desde la perspectiva de planificación de tareas con prioridad fija apropiativa para sistemas monoprocesador. Este método sería adaptado para la programación de mensajes CAN, que no incluye la posibilidad de apropiación del canal de comunicaciones compartido. A su vez, también se complementa con un algoritmo de asignación de prioridades óptimo aplicable a los sistemas no apropiativos. Se muestran resultados sobre una implementación de una red real para verificar experimentalmente estos tiempos. Este análisis garantiza tiempos conocidos y correctos, los mensajes y señales serán distribuidos dentro de los plazos, requisito fundamental para un sistema de tiempo real.

Palabras Clave: Bus CAN, Comunicaciones de Tiempo Real, Sistemas de Tiempo Real.

1 Introducción

El desarrollo inicial de un análisis de tiempos CAN se basa en [5], directamente sobre la planificación de tareas con prioridad fija apropiativa para sistemas de un único procesador. Este método es adaptado para la programación de mensajes CAN y se complementa con lo investigado en [3], que mostró cómo un algoritmo de asignación de prioridades óptimo introducido en [1] es aplicable a los sistemas no apropiativos. Todos estos desarrollos se pueden verificar de manera experimental al menos en parte construyendo una red CAN real sobre la cual tomar tiempos de comunicaciones, evaluando los plazos de tiempo real que se pueden cumplir/asegurar.

Los buses definidos en CAN se arbitran utilizando prioridades, y una vez que se obtiene el acceso al bus, el mensaje se transmite completo, sin posibilidad de apropiatividad en el caso de la llegada de un mensaje de mayor prioridad mientras se está transmitiendo el mensaje para el que se accedió al canal de comunicaciones. Para garantizar que se transmitan los mensajes de más alta prioridad con baja latencia, se debe tratar la interferencia, o en realidad el retraso producido por parte de señales menos urgentes, con menor prioridad. Por otro lado, también se debe tener en cuenta

[1]Comisión de Investigaciones Científicas de la Prov. de Buenos Aires.

en general que no solamente se tiene la posibilidad de tiempo de espera en estos casos de prioridades invertidas (un mensaje de menor prioridad está usando el bus mientras uno de mayor prioridad está esperando para acceder), sino que todos los mensajes deben esperar el acceso al medio de comunicación cuando hay mensajes de mayor prioridad, que accederán previamente. Todo el desarrollo del trabajo se enfocará en el comportamiento de la transmisión de Data Frames del estándar CAN.
Los mensajes de todos los nodos en la red que acceden al medio de transmisión por un controlador comparten una única cola basada en prioridades y son enviados al bus acorde a una planificación no apropiativa de prioridad fija. CAN no tiene un reloj único/global para todos los controladores, por lo tanto cada controlador CAN tiene su propia señal de reloj, que con un grado de tolerancia especificado por el protocolo, puede comunicarse con el resto de los nodos. Por ello, los nodos se resincronizan al momento de la transmisión de un mensaje en el flanco ascendente del bit de comienzo de una transmisión, causado por quien haya comenzado a transmitir primero. Los nodos pueden comenzar una transmisión cuando el bus se encuentra desocupado por un espacio de 3 bits interframe. El nodo comienza a transmitir con un bit dominante ('0') lo que provoca que el resto de los nodos se sincronicen con el flanco ascendente de este bit y se vuelvan receptores.

Cualquier mensaje que esté listo para ser transmitido deberá esperar a que se inicie otro ciclo de arbitraje en el momento en el cual el bus quede nuevamente desocupado. Por definición, el protocolo CAN necesita que el transmisor inserte un bit con polaridad opuesta cada 5 bits de la misma polaridad. Este proceso es conocido como "bit stuffing" y es luego invertido por el receptor. En los primeros trabajos, como en [5] no se tuvo en cuenta este proceso, pero luego en [2], se agrega este proceso al modelado de tiempos para la transmisión de los mensajes. En el peor caso, se debe agregar un bit de stuffing cada 4 bits del mensaje y, por lo tanto, el tiempo máximo de transmisión estaría dado por C_m en la Eq. (1)

$$C_m = \left(g + 8s_m + 13 + \left\lfloor \frac{g + 8s_m - 1}{4} \right\rfloor \right) \tau_{\text{bit}} \quad (1)$$

donde s_m es la cantidad de bytes de datos del mensaje, g es una constante, 34 para el formato estándar de identificador o 54 para el formato extendido, y τ_{bit} es el tiempo de transmisión de un bit.

2 Tiempos: Esperas, Máximo, Errores

Si bien los tipos de mensajes son dependientes de la aplicación, los tipos de mensajes son predefinidos para la simulación y para la implementación. De hecho, cada mensaje m tiene un identificador y prioridad únicos y fijos, se puede decir tanto mensaje m como mensaje de prioridad m. Cada mensaje se encola por una rutina de software, por el flujo normal del programa o por ejecución de una interrupción en el host. Esta tarea necesita de un tiempo para ejecutarse entre 0 y J_m, queuing jitter del mensaje, y es heredado desde el tiempo de respuesta de la tarea, incluyendo la demora por polling [5]. Los eventos que disparan el encolamiento ocurren con un mínimo de periodicidad de T_m, el periodo del mensaje, siendo soportados los tipos de eventos: a)

Los que ocurren estrictamente con un periodo de T_m, b) Los que ocurren esporádicamente con un mínimo de separación de T_m, c) Los que ocurren solo cuando el sistema es reiniciado. Cada mensaje tiene un *hard deadline* D_m: tiempo máximo desde el evento inicial hasta la transmisión del mensaje.

Se puede definir el peor caso de tiempo de respuesta R_m de un mensaje como el mayor tiempo desde que ocurre el evento de inicio hasta que el mensaje comienza a ser recibido por los nodos que lo requieran. Un mensaje se puede llamar "planificable" (schedulable) si y sólo si su peor caso de tiempo de respuesta es menor o igual a D_m ($R_m \leq D_m$). El sistema es "planificable" (schedulable) si y sólo si todos los mensajes son planificables. R_m es, básicamente, la suma de dos tiempos: 1) El retardo W_m (también llamado *Window time*), que es el mayor tiempo de permanencia del mensaje en el controlador CAN, y 2) El peor tiempo de transmisión, C_m. A su vez, el retardo del tiempo de ventana W_m se compone de: 1) Tiempo de bloqueo, B_m, dado por mensajes de menor prioridad en proceso de transmisión, y 2) Tiempo de interferencia, donde un mensaje con prioridad más alta gana el arbitraje y se transmite en vez del mensaje m.

El tiempo de bloqueo es relativamente sencillo de calcular, como $B_m = \max(C_k)$ con $k \in lp(m)$, donde $lp(m)$ es el conjunto de mensajes con menor prioridad que m. El tiempo de interferencia está muy relacionado al concepto de "período ocupado" (busy period) introducido en [4], donde se definen a estos períodos ocupados de nivel i como el intervalo de tiempo [a, b] dentro del cual las tareas con una prioridad i o mayor son procesados dentro de este intervalo y no se tienen tareas de esta prioridad en los intervalos (a - t, a) y (b, b + t) siendo un tiempo $t > 0$. Adaptando esta definición al protocolo CAN, se tiene que un período ocupado de prioridad m. El período se inicia en un tiempo T_s donde un mensaje de prioridad m o mayor es encolado listo para transmitir, y no hay mensajes de prioridad m o mayor que estén esperando a ser transmitidos, encolados estrictamente antes de T_s. Es un intervalo continuo de tiempo, durante el cual cualquier mensaje de prioridad menor que m está imposibilitado de comenzar una transmisión y ganar el arbitraje. Finalmente, el período concluye en un tiempo T_e, cuando el bus pasa a estar listo para el siguiente ciclo de transmisión y arbitraje y aún no hay mensajes de prioridad m o superior esperando a ser transmitidos que hayan sido encolados estrictamente antes del tiempo T_e.

La característica clave del periodo ocupado es que todos los mensajes de prioridad m o superior encolados estrictamente antes de la finalización son transmitidos mientras dure el periodo. En términos matemáticos, los períodos ocupados pueden ser vistos como un intervalo [ts, Te). En consecuencia, el final de un periodo ocupado puede coincidir con el comienzo de otro periodo ocupado [4]. Se puede calcular tiempo ocupado como el máximo de

$$t_m^{n+1} = B + \sum_{\forall j \in hp(m)} \left\lceil \frac{t_m^n + J_j + \tau_{bit}}{T_j} \right\rceil C_j \qquad (2)$$

donde $hp(m)$ es el conjunto de mensajes con prioridad más alta que m, y J_j y T_j se relacionan con el jitter y la periodicidad de los mensajes respectivamente, tal como se explica anteriormente. Se puede observar que el tiempo ocupado toma una forma monótona creciente, iniciando con el valor B_m e incrementándose hasta alcanzar uno

de los siguientes casos: a) $t^n_m + C_m > D_m$, en el que se tiene un mensaje que no es válido por superar su deadline, o b) $t^{n+1}_m = t^n_m$, donde se tiene el peor tiempo de respuesta de la primera instancia del mensaje en el período ocupado que está dado por $t^n_m + C_m$.

Al considerar los errores y los tiempos de recuperación relacionado, un nodo al detectar un error comienza el proceso de recuperación transmitiendo un indicador de error. El proceso de resincronización toma al menos 29 tiempos de transmisión de 1 bit. Se define una función que represente este error, E(t), como el mayor tiempo que necesario para la recuperación de los errores en cualquier intervalo de duración t [4]. La sobrecarga que involucran los errores en cualquier intervalo de duración t sería

$$\left(n_{burst} + \left\lceil \frac{t}{T_{error}} \right\rceil\right) C_{overhead}$$

Donde $C_{overhead}$ es el mayor tiempo que demora el manejo de un error y el reenvío del mensaje más largo transmitido en un intervalo, el peor caso de latencia, dado por R_m. El mayor tiempo de teniendo en cuenta los errores entonces sería

$$C_{overhead,m} = \max_{\forall j \in (hp(i) \cup \{i\})} (C_j) + 29\tau_{bit}$$

Por lo que finalmente se puede definir E_m como

$$E_m = \left(n_{burst} + \left\lceil \frac{t_m + C_m}{T_{error}} \right\rceil\right) C_{overhead,m}$$

Teniendo en cuenta esta componente aportada por los errores se llega a una ecuación con la cual analizar los peores casos en ambientes con ruido:

$$t_m = B_m + E_m + \sum_{\forall j \in hp(m)} \left\lceil \frac{t_m + J_j + \tau_{bit}}{T_j} \right\rceil C_j \qquad (3)$$

3 Benchmark Específico

La Sociedad de Ingenieros Automotrices (SAE: Society of Automotive Engineers) proporciona un banco de datos de prueba utilizado para evaluar diferentes tecnologías de comunicación de señales que son enviadas entre siete subsistemas diferentes en un prototipo de automóvil eléctrico [5]. Este banco de pruebas es usado para mostrar un caso de uso "real" para las ecuaciones y modelos que anteriores. Entre los siete subsistemas que generan señales están, por ejemplo, los frenos, el propio conductor, el motor, etc. En total se tienen 53 señales, caracterizadas no solo por el subsistema que las generan sino por la cantidad de bits de información, si son periódicas o esporádicas, jitter, etc. Un mensaje periódico tiene un periodo fijo de tiempo entre invocaciones y el esporádico un requisito de latencia impuesto por la aplicación. Se considera como válido el mensaje que cumple con esta latencia.

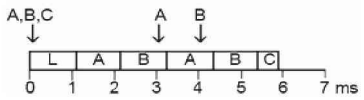
El análisis presentado es válido para cualquier orden del conjunto de mensajes. Es importante seleccionar el orden y las prioridades correctas a la hora de obtener un sistema planificable, y de esta manera maximizar la robustez del sistema frente a

errores. En [5] se propone una asignación de prioridades "por deadline" (*deadline monotonic priority assignment*), asignando a los de menor diferencia la mayor prioridad que solo es óptima para sistemas cuya programación de tareas sea de prioridad fija apropiativa, asumiendo plazos no mayores a sus periodos. Que no sea óptimo no significa que no sea utilizable, solo que en caso de tener tiempos más críticos se debería elegir otra opción. Por ejemplo, según la Tabla 1, y asumiendo jitter 0 y tasa de transferencia de 125 Kb/s, el cálculo cumple con lo planteado en el análisis, donde se tenía $R_m < D_m$. Sin embargo, se puede ver un contraejemplo usando

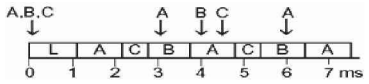
Tabla 1. Ejemplo de Caracterización de Mensajes.

Mens.	Periodo.	Deadline	Bytes	T transm.	T transm.
A	3.0 ms	3.0 ms	8	1.08 ms	2.16 ms
B	4.0 ms	4.0 ms	8	1.08 ms	2.16 ms
C	4.5 ms	4.5 ms	1	0.53 ms	3.24 ms

la misma Tabla 1, en el que no siempre todos los mensajes pueden ser atendidos antes del deadline:



en el inicio, los tres mensajes A, B, y C esperan que el mensaje L de prioridad más baja termine su transmisión. Luego de esto, A y B ganan sucesivamente el arbitraje frente a C que no llega a transmitirse antes de su deadline de 4.5 ms. Esta situación pretende ser la óptima, y según esto no debería haber otro orden de prioridad que cumpla con lo buscado. Sin embargo, si se tiene en cuenta el orden A, C y B se obtienen los peores tiempos de respuesta, con $R_A = 2.16$, $R_C = 2.68$, y $R_B = 3.76$. Con este orden de prioridad y estos tiempos de respuesta, todos los mensajes alcanzan sus deadlines:



Es decir que dando la mayor prioridad al mensaje más corto, los tres mensajes comienzan su transmisión dentro de los 3ms de ser encolados. Ninguno de ellos estará sujeto a interferencia de una segunda instancia del mensaje A y consecuentemente a una segunda instancia del mensaje B. Para este tipo de situaciones, en [3] se muestra que es aplicable a los sistemas no apropiativos un algoritmo de asignación de prioridades óptimo que se describe en [1]. En general, el algoritmo de [1] es aplicable siempre y cuando el peor caso de tiempo de respuesta de un mensaje: a) no dependa de un ordenamiento de los mensajes de más alta prioridad, y b) no es de mayor tamaño si se le da una prioridad mayor.

La longitud de la cola de demoras y la longitud del periodo ocupado no dependen de un orden específico de los mensajes de alta prioridad. El término bloqueante B_m puede hacerse mayor al incrementar la prioridad, pero a su vez es contrarrestado por un decremento de la interferencia. Es posible encontrar un ordenamiento óptimo

siguiendo el algoritmo de dado en [1]:

Algoritmo de asignación óptima de prioridades( )

```
{
  Para cada nivel de prioridad, comenzando desde el más bajo
  {
     Para cada mensaje no asignado m
     {
        Si m es planificable con esta prioridad
           asignar a m esta prioridad
        else
           return El sistema no es planificable
     }
  }
  return El sistema es planificable
}
```

Para n mensajes, este algoritmo realiza como máximo n(n-1)/2 evaluaciones y garantiza una asignación de prioridades planificables si es que existe. Se debe tener en cuenta que el algoritmo no especifica un orden en el que los mensajes deberían ser analizados en cada nivel de prioridad. Este orden influye en gran medida en la asignación de la prioridad elegida si hay más de un orden planificable. Una mala elección de pedido inicial puede resultar en una asignación de prioridades que deje al sistema como planificable, sin ninguna seguridad de que se haya alcanzado un grado óptimo. Teniendo los datos presentados por la SAE y los modelos presentados, se pueden aplicar los cálculos propuestos por el análisis. Sin embargo, estos pasos pueden ser aplicados a cualquier conjunto de mensajes CAN siguiendo los siguientes pasos:

1. Obtención de las características de los mensajes: tamaño, identificador, velocidad del bus, etc.
2. Ordenar los mensajes por algún criterio, sea el de dado en [5] o [1].
3. Aplicar el cálculo propuesto a cada mensaje.

Para caso de la SAE se usará como guía la propuesta dada en [5], el método "D - J" en los que asigna mayor prioridad a los que tienen menor margen de espera (es decir, un valor de D - J más pequeño). Los valores de la Tabla 2 muestran el resultado.

Un punto importante a tener en cuenta es la periodicidad al momento de ordenar los mensajes. En este caso, el mensaje 14 es esporádico, y se puede dar como mínimo una vez cada 50ms, pero su deadline es de 5ms. Teniendo en cuenta esto, y que tal como lo indica la Tabla 2 es una señal correspondiente a la acción del freno, es natural que aparezca primero en la tabla. Para los mensajes periódicos, se tiene que el deadline es igual al tiempo del periodo, ya que de no ser atendido antes de llegar este punto el mensaje es sobrescrito inevitablemente. a partir del quinto mensaje analizado (el número de señal 8) el tiempo se dispara y no llegamos a transmitir el mensaje antes de alcanzar el deadline de 5ms. Para estos casos no nos quedan más opciones que subir la velocidad en la que operamos sobre el bus, o utilizar la técnica de "*piggyback*".

Utilizar piggyback en CAN consiste en agrupar los mensajes que provienen del mismo origen, en un mensaje único. Por ejemplo, el subsistema que representa a la batería envía cuatro señales de un byte cada 100ms (las señales 1, 2, 4 y 6). Estas pueden ser representadas como un único mensaje de 4 bytes. Esto reduce la

utilización del bus ahorrando la sobrecarga de tres mensajes. La transmisión de un byte de datos puede llegar a requerir la transmisión de hasta 63 bits. Esta técnica se puede extender a las señales que poseen distintos periodos. Por ejemplo, las señales 29, 30 y 32 pueden ser transmitidas en un mensaje con un periodo de 5ms, mientras que dos de las señales tienen un periodo de 10ms (por cada segundo mensaje de la primera, se envían las dos señales de las restantes, cuando no son enviadas se puede enviar un valor nulo en su lugar).

Tabla 2. Cálculo Aplicado al Benchmark SAE.

Signal Number	Signal Description	Size/bits	J/ms	T/ms	Periodic / Sporadic	D/ms	From	Rm para 125Kbs
14	Hi&Lo Contactor Open/Close	4	0,1	50	S	5	Battery	1,776
9	Brake Pressure, Line	8	0,2	5	P	5	Brakes	2,816
49	Processed Motor Speed	8	0,2	5	P	5	I/M C	3,856
42	Torque Command	8	0,2	5	P	5	V/C	4,896
8	Brake Pressure, Master Cylinder	8	0,1	5	P	5	Brakes	5,936
7	Accelerator Position	8	0,1	5	P	5	Driver	11,136
43	Torque Measured	1	0,1	5	P	5	I/M C	17,880
11	Transaction Clutch Line Pressure	8	0,1	5	P	5	Trans	25,632
32	Clutch Pressure Control	8	0,1	5	P	5	V/C	36,536
29	High Contactor Control	8	0,3	10	P	10	V/C	65,088
30	Low Contactor Control	8	0,4	10	P	10	V/C	112,024
53	Main Contactor Acknowledge	1	1,5	50	S	20	V/C	207,744
48	Shift in Progress	1	1,4	50	S	20	V/C	378,352
46	Idle	1	1,3	50	S	20	V/C	686,120
44	FWD/REV	1	1,2	50	S	20	V/C	1250,728
40	Key Switch	1	1,1	50	S	20	V/C	2288,960
39	Warning Lights	1	1	50	S	20	V/C	4192,712
27	SOC Reset	1	0,9	50	S	20	Driver	7721,760
38	Backup Alarm	7	0,9	50	S	20	V/C	14297,048
37	Brake Solenoid	1	0,8	50	S	20	V/C	26734,536
52	Status/Malfunction (TBD)	8	0,8	50	S	20	I/M C	50253,808

Tabla 3. Análisis de Piggybacking.

Signal N°s	*Size /bytes*	*J /ms*	*T /ms*	*D /ms*	*R (125Kbit/s)*
14	1	0.1	50.0	5.0	1.544
8,9	2	0.1	5.0	5.0	2.128
7	1	0.1	5.0	5.0	2.632
43,49	2	0.1	5.0	5.0	3.216
11	1	0.1	5.0	5.0	3.720
32,41	2	0.1	5.0	5.0	4.304
31,34,35,37,38,39,40,44,46,48,53	6	0.2	10.0	10.0	5.192
23,24,25,28	1	0.2	10.0	10.0	8.456
15,16,17,19,20,2226,27	2	0.2	10.0	10.0	9.040
41,43,45,47,49,50,51,52	3	0.2	10.0	10.0	9.696
18	1	0.2	50.0	20.0	10.200
1,2,4,6	4	0.3	100.0	100.0	19.088
12	1	0.3	100.0	100.0	19.592
10	1	0.2	100.0	100.0	20.096
3,5,13	3	0.4	1000.0	1000.0	28.904
21	1	0.3	1000.0	1000.0	29.408
33,36	1	0.3	1000.0	1000.0	29.912

La técnica piggyback también se puede aplicar a señales que no son generadas al mismo tiempo (como puede ser el caso de señales esporádicas). Dichas señales pueden ser enviadas en un mensaje "servidor" donde la estación que envía el mensaje verifica constantemente la aparición de la señal esporádica antes de encolar el mensaje. Con este enfoque, una señal esporádica puede ser demorada por no más de un tiempo dado por el periodo de polling, más el peor tiempo de latencia del mensaje "servidor". Si se tuviera que aplicar la técnica en un mensaje que tiene como deadline

20ms, un mensaje servidor con un periodo de polling de 15ms y un peor caso de latencia de 5ms sería suficiente. Para usar esta técnica se deberían repetir los pasos anteriores, pero teniendo en cuenta que se busca que las distintas señales tengan un tamaño mínimo. Una vez esto resuelto, se deben agrupar los distintos mensajes para finalmente volver a calcular los tiempos con los nuevos mensajes. En el caso de la Tabla 2, se puede llegar a los valores de la Tabla 3, donde se logran transmitir todos los mensajes comprometidos antes de alcanzar los distintos deadlines. Esto es debido a que se reduce la utilización del bus por debajo del 100%. de transmisión de un bit.

4 Ambiente de Experimentación y Validación

La Fig. 1-a) muestra de manera esquemática la red CAN construida para validar el análisis anterior en una red real, aunque claramente de experimentación. La Fig. 1-b) muestra la red en funcionamiento, en particular la salida serie con registros de tiempo de la red en funcionamiento. La Fig. 2 sigue el mismo esquema de construcción de la red CAN experimental, incluyendo la placa Kinetis K70.

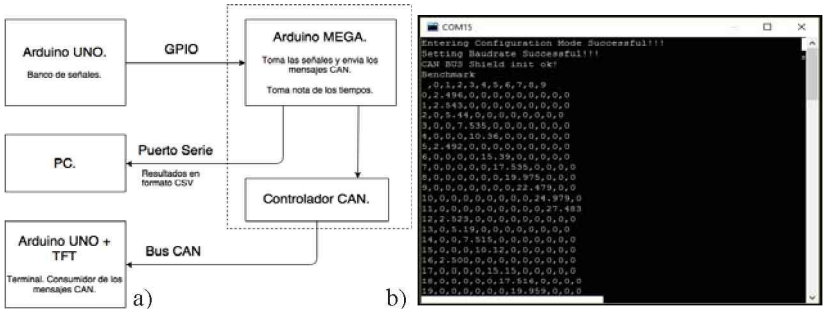


Fig. 1. Ambiente Experimental de CAN: a) Esquema del Hardware, b) Mediciones.

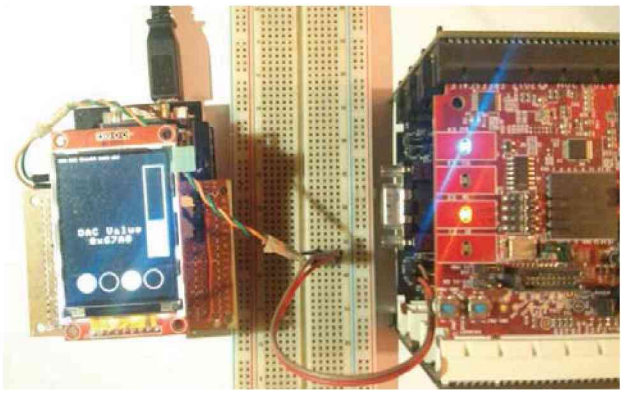


Fig. 2. Variante de Implementación de Red CAN con Kinetis K70.

Si bien el segmento presentado en la Fig. 3 corresponde a un lapso de tiempo muy reducido, permite verificar que en la red real construida el envío de los mensajes se da como en "escalera". De esta manera, se corrobora la ecuación en la que se basa el análisis de tiempos de espera, basado en el valor conocido como "periodo de ventana". Tal como se puede derivar de la ecuación, que utiliza los peores tiempos de los mensajes con mayor prioridad.

	A	B	C	D	E	F	G	H	I	J	K
1		0	1	2	3	4	5	6	7	8	9
2	0	2,576	0	0	0	0	0	0	0	0	0
3	1	2,579	0	0	0	0	0	0	0	0	0
4	2	2,584	0	0	0	0	0	0	0	0	0
5	3	0	2,66	0	0	0	0	0	0	0	0
6	4	0	0	5,248	0	0	0	0	0	0	0
7	5	0	0	0	7,84	0	0	0	0	0	0
8	6	0	0	0	0	10,432	0	0	0	0	0
9	7	0	0	0	0	0	13,27	0	0	0	0
10	8	0	0	0	0	0	0	15,564	0	0	0
11	9	0	0	0	0	0	0	0	18,163	0	0
12	10	0	0	0	0	0	0	0	0	0	20,752

Fig. 3. Tiempos de Esperas de los Mensajes Medidos en la Red CAN.

En la Fig. 4, se puede observar el gráfico resultado a partir de las métricas obtenidas a través del benchmark sobre la red real construida. En este gráfico se puede identificar la cadencia con la que los mensajes van llegando al controlador CAN, y cómo el tiempo de cada mensaje depende de los mensajes de mayor prioridad que se procesan con anterioridad.

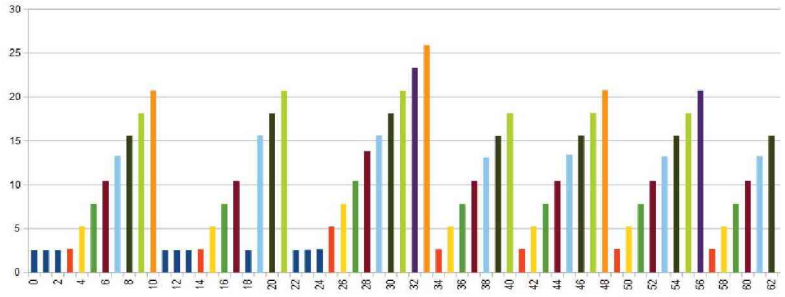


Fig. 4. Periodos Ocupados Medidos en la Red CAN.

Cada barra vertical de la Fig. 4 identifica un mensaje o instancia de mensaje en particular (el tipo de señal se identifica con un color), numerándolos desde 0 en adelante (sobre el eje x) y la altura de la cada barra es directamente el tiempo del mensaje desde que llega al controlador CAN hasta que es recibido en destino. Interpretando las primeras 10 barras de la Fig. 3 se puede identificar que:

- Los mensajes CAN 0 al 2 pertenecen a una misma señal 0, que tiene periodo de 30ms, por lo que el emisor puede enviar los 3 mensajes antes de que llegue la señal 1 que tiene un periodo de 100ms.
- Luego de esto, en un mismo instante (a los 100ms) llegan datos de las señales 1 a la 8. Estas entran todas al mismo tiempo, por lo que se tienen tiempos

incrementales de los mensajes 3 y 10 del gráfico, de acuerdo a la forma en que se envían los mensajes CAN teniendo en cuenta las prioridades.

- El envío de los mensajes que llegaron en el mismo instante suceden en menos de 30ms y luego llega un nuevo mensaje de la señal 0. Es importante notar que ninguno de los mensajes tuvo una latencia superior a la peor calculada anteriormente por medio de los valores de la tabla, por lo que se puede decir que se prueba el cálculo y puesta en marcha de este tipo de análisis de tiempos de manera satisfactoria.

5 Conclusiones y Trabajo Futuro

Se pudo llevar a cabo la corroboración experimental del análisis planteado en trabajos anteriores sobre los tiempos de latencia y respuesta en la transmisión de mensajes utilizando el protocolo CAN. Se ha explorado la representación matemática de cada uno de los fenómenos presentes en el bus buscando las herramientas necesarias para el análisis de un conjunto de señales de un sistema hipotético, tomando como base el benchmark SAE. Este enfoque fue y es utilizado por los fabricantes de automóviles y maquinaria a la hora de plantear sus sistemas basados en CAN, por lo que se deja presente en este trabajo un análisis que si bien es útil por sí mismo, a su vez también permite nuevas investigaciones y enfoques. Se implementó y validó en una prueba sobre un sistema real, construyendo no solamente el hardware sobre el cual llevar a cabo la experimentación sino incluyendo también la instrumentación necesaria para la recolección de los datos a contrastar con el modelo matemático.

Entre los trabajos futuros sin lugar a dudas se debe considerar un estudio cuidadoso para incrementar el ambiente de hardware de experimentación, principalmente en lo que se refiere a contar con una red CAN más cercana a las reales en cuanto a cantidad de controladores. De manera análoga se debe determinar cuidadosamente el ancho de banda del bus y la cantidad de fuentes de mensajes (tipos de señales involucradas) que cada controlador CAN debe administrar/manejar en el acceso al bus CAN.

Referencias

1. Audsley, N. C., "Optimal priority assignment and feasibility of static priority tasks with arbitrary start times", 1991, http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.41.6673.
2. Davis, R., Burns, A., Brill, R., Lukkien, J. J. "Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised", Springer, Real-Time Syst 35: 239, 2007. doi:10.1007/s11241-007-9012-7.
3. George, L., Rivierre, N., Spuri, M., "Preemptive and Non-Preemptive Real-Time Uni-Processor Scheduling", Rapport de recherche n°2966, INRIA Rocquencourt, 1996.
4. Lehoczky, J. P., "Fixed Priority Scheduling of Periodic Task Sets with Arbitrary Deadlines", Proceedings., 11th Real-Time Systems Symposium, pp. 201-209, 1990.
5. Tindell, K., Burns, A., Wellings, A. J., "Calculating controller area network (CAN) messages response times", Control Eng. Practice, Vol.3 No. 8, 00. 1163-1169, Elsevier Science Ltd., 1995.